

# Itinéraire entre deux stations du métro parisien

*Kim Antunez et Alain Quartier-la-Tente*

*07/01/2020 - 15h30 à 15h45*

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Prise en main du programme</b>	<b>1</b>
<b>3</b>	<b>Description des données</b>	<b>1</b>
3.1	Description des données à disposition . . . . .	1
3.2	Difficultés et solutions adoptées . . . . .	2
<b>4</b>	<b>Description des classes</b>	<b>4</b>
<b>5</b>	<b>Description de l'algorithme</b>	<b>4</b>
<b>6</b>	<b>Pistes d'amélioration</b>	<b>4</b>
<b>7</b>	<b>Lignes de métro</b>	<b>4</b>

## 1 Introduction

Ce rapport décrit le projet C++ de Kim Antunez et d'Alain Quartier-la-Tente dont l'objectif est de permettre à l'utilisateur d'obtenir un chemin entre deux stations de métro selon deux critères : le plus court chemin ou le chemin avec le moins de correspondances. L'ensemble des données et des codes utilisés sont disponibles sous [https://github.com/AQLT/Metro\\_Cpp](https://github.com/AQLT/Metro_Cpp), la section 2 décrivant comment prendre en main et utiliser l'application. Les données utilisées sont les données du métro parisien fournies par la RATP (section 3), l'implémentation des classes est décrite dans la section 4, l'algorithme utilisé pour calculer les chemins est l'algorithme de Dijkstra (section 5). Enfin, les pistes d'amélioration sont décrites dans la partie 6.

Pour mieux se retrouver dans le réseau parisien, le plan de l'ensemble des lignes a été rajouté dans la section 7.

## 2 Prise en main du programme

## 3 Description des données

### 3.1 Description des données à disposition

Toutes les données utilisées dans ce projet sont issues de la RATP de la base `offre-transport-de-la-ratp-format-gtfs` (<https://datarotp.opendatasoft.com/explore/dataset/offre-transport-de-la-ratp-format-gtfs/information/>). Ces données sont au format *General Transit Feed Specification* (GTFS) qui est un format standardisé pour publier un réseau de transport en commun (horaires, informations géographiques, etc.).

Pour les données de la RATP sont disponibles sous deux formes d'archives :

- Une archive avec des fichiers GTFS réparties par lignes ;
- Une archive avec des fichiers GTFS pour l'ensemble du réseau (métro, bus, tram et RER).

C'est cette première archive que nous avons utilisé pour nous restreindre plus facilement aux données des lignes de métro uniquement. Nous avons stockés les données utilisées ici : [https://github.com/AQLT/Metro\\_Cpp/tree/master/Data](https://github.com/AQLT/Metro_Cpp/tree/master/Data).

Pour chaque ligne, les données sont stockées dans différents fichiers :

1. `routes.txt` : définit les itinéraires des transports en commun → données **utilisées** avec le fichier `trips.txt` pour identifier l'ordre de passage à chaque arrêt pour les lignes aller et retour.
2. `stops.txt` : définit l'ensemble des arrêts où les usages peuvent monter ou descendre, avec le nom de l'arrêt, l'adresse et les coordonnées GPS → données **utilisées** dans ce projet pour définir l'ensemble des arrêts.
3. `stop_times.txt` : définit, pour chaque trajet et pour chaque arrêt, les heures d'arrivée et de départ du métro → données **utilisées** avec le fichier `trip.txt` pour connaître le temps de trajet entre deux stations de la même ligne.
4. `transfers.txt` : définit les règles de liaison aux pôles de correspondance entre des itinéraires → données **utilisées** dans ce projet pour connaître les correspondances et les temps de correspondance entre les lignes.
5. `trips.txt` : définit l'ensemble des trajets pour chaque ligne (i.e. : tous les trajets prévus dans la journée) → données **utilisées** avec le fichier `routes.txt` et `stop_times.txt` pour identifier pour chaque ligne l'ordre de passage et le temps de trajet entre chaque arrêt <sup>1</sup>.
6. `calendar_dates.txt` : définit les exceptions pour les services définis dans le fichier `calendar.txt` → données **non utilisées** dans ce projet.
7. `calendar.txt` : définit les dates auxquelles le service est disponible pour des itinéraires spécifiques selon un calendrier hebdomadaire. Ce fichier spécifie les dates de début et de fin du service, ainsi que les jours de la semaine où le service est disponible → données **non utilisées** dans ce projet.
8. `agency.txt` : définit une ou plusieurs agences de transports publics dont les services sont représentés dans l'ensemble de données → données **non utilisées** dans ce projet.

Plus d'informations sur les données GTFS sont disponibles sur le site de Google : <https://developers.google.com/transit/gtfs/reference/>.

### 3.2 Difficultés et solutions adoptées

Chaque arrêt est défini par une identifiant unique et cet identifiant est différent pour chaque ligne et pour chaque route (aller et retour) sans qu'aucun temps de correspondance n'ait été défini dans les données. Certains itinéraires étaient donc impossibles à calculer : par exemple, si l'on est sur l'arrêt de métro Gaîté sur la ligne 13 direction Châtillon-Montrouge on ne peut rejoindre l'arrêt Montparnasse.

→ **Solution adoptée** : ajouter un temps de transfert égal à 0 pour chaque arrêt pour passer d'une ligne aller à la ligne retour et inversement. Ainsi, le temps de transfert est nul pour passer de l'arrêt de métro Gaîté sur la ligne 13 direction Châtillon-Montrouge à l'arrêt de métro Gaîté sur la ligne 13 direction Saint-Denis/Les Courtilles. En revanche cela rend le calcul du temps de trajet moins fiable pour deux raisons :

1. Le temps de transfert entre deux lignes n'est pas le même en fonction de la direction que l'on prend.
2. Pour faire certains itinéraires, il est nécessaire de changer de direction tout en restant sur la même ligne (par exemple sur la ligne 13 passer de Guy Môquet à Brochant il faut aller jusqu'à l'arrêt La Fourche et changer de direction). Puisque nous n'avons pas pris en compte de temps d'attente moyen d'un métro, le temps du trajet est sous-estimé.

Cette façon de coder les arrêts implique que certains arrêts ne sont associés qu'à une ligne (aller ou retour) alors que d'autres sont associés à deux lignes (par exemple sur la ligne 13 avec 2 lignes aller ou 2 lignes retour).

---

1. Dans ce projet nous ne prenons pas en compte l'heure à laquelle la recherche d'itinéraire a été faite : pour chaque "route" un seul "trip" a donc été utilisé

Plusieurs incohérences sur les trajets ont également été trouvées :

- Le fichier `routes.txt` ne permettait pas toujours de bien identifier les lignes aller et retour. En effet, pour certaines lignes de métro, pour le même identifiant de “trip” et pour une “route” fixée il pouvait y avoir au même horaire un départ d’un terminus pour une direction et du terminus opposé pour l’autre direction. Cela devrait normalement être impossible puisque la “route” permet d’identifier la directement. Ce problème affecté les lignes 1, 4, 7, 7B et 13 : un travail manuel a donc été fait pour identifier correctement les lignes aller et retour.
- Certaines “routes” de la base de données n’existent pas en vrai. C’est le cas d’une des routes de la ligne 10 “BOULOGNE - PONT DE SAINT CLOUD <-> GARE D’AUSTERLITZ) - Aller” qui partirait de l’arrêt Porte d’Auteuil pour ensuite aller à l’arrêt Michel-Ange Molitor et continuer direction Gare D’austerlitz (alors que depuis Porte d’Auteuil la seule direction possible est Boulogne). Cette “route” n’a pas été considéré dans notre algorithme.

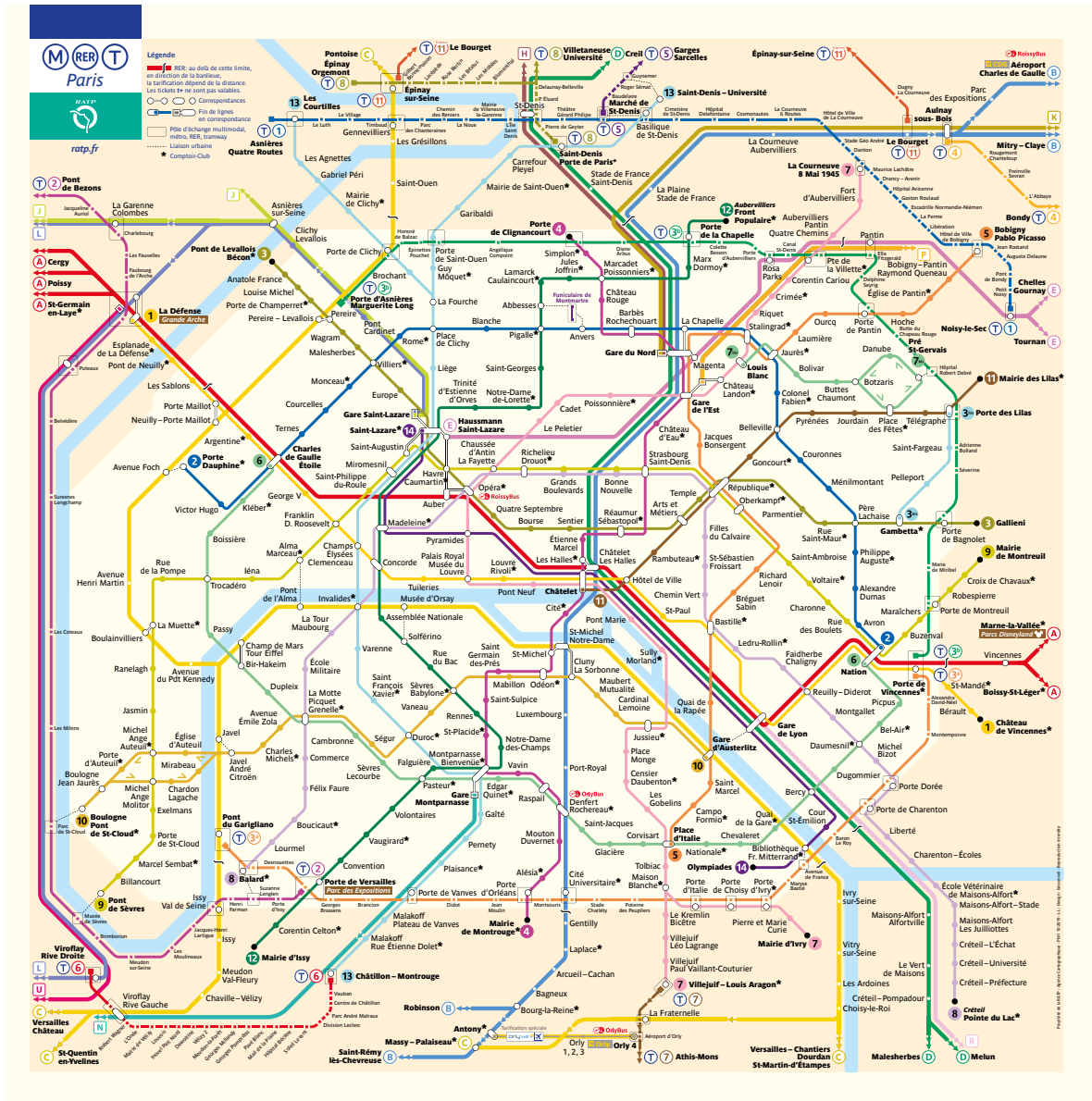
Pour simplifier le chargement des données en C++, nous avons pré-traité les données dans le logiciel statistique

:

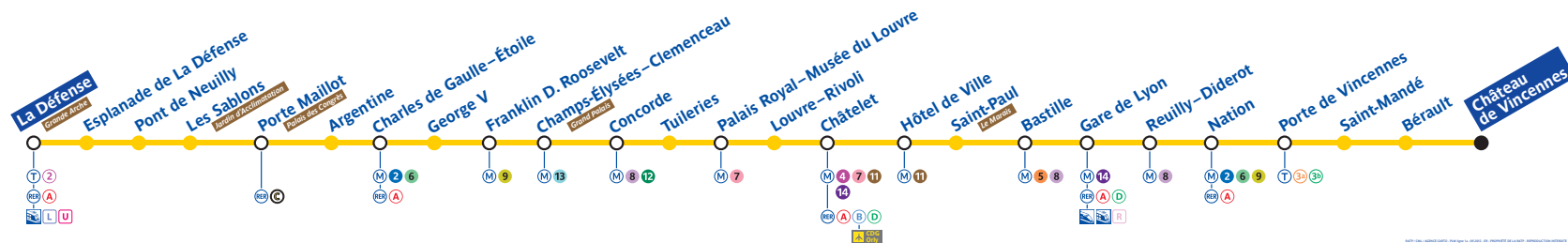
- Pour chaque “route” nous avons créé un fichier avec :
  - Le fichier `stops.txt` où on a enlevé les accents (permet de créer l’ensemble des arrêts) ;
  - L’ensemble des arrêts de manière ordonnée ainsi que le nom de la route (permet de créer toutes les lignes, d’identifier les arrêts traversés ainsi que l’ordre de passage).
- Deux matrices carrées ayant autant de colonnes que d’identifiants d’arrêts :
  - `voisins_type.txt` : la coordonnée (i,j) vaut -1 si les deux arrêts ne sont pas directement connectés, 0 si les deux arrêts sont connectés et sont sur la même ligne (i.e. : si ce sont des arrêts voisins) et 1 si les deux arrêts sont connectés mais sur deux lignes différentes (par exemple entre la ligne 4 et la ligne 13 à l’arrêt Montparnasse).
  - `voisins.txt` : la coordonnée (i,j) correspond au temps nécessaire pour aller directement l’arrêt i à l’arrêt j (avec une valeur égale à -1 s’il n’y a pas de correspondance directe possible).

L’ensemble de ces nouveaux fichiers sont stockés sous [https://github.com/AQLT/Metro\\_Cpp/tree/master/Data%20projet](https://github.com/AQLT/Metro_Cpp/tree/master/Data%20projet).

- 4 Description des classes
- 5 Description de l'algorithme
- 6 Pistes d'amélioration
- 7 Lignes de métro

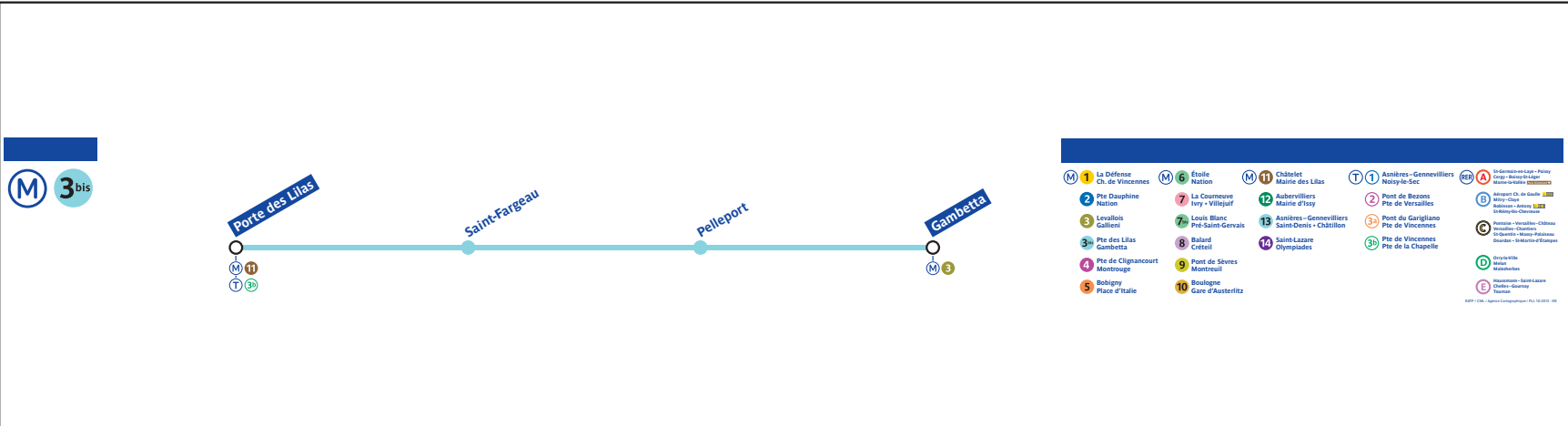
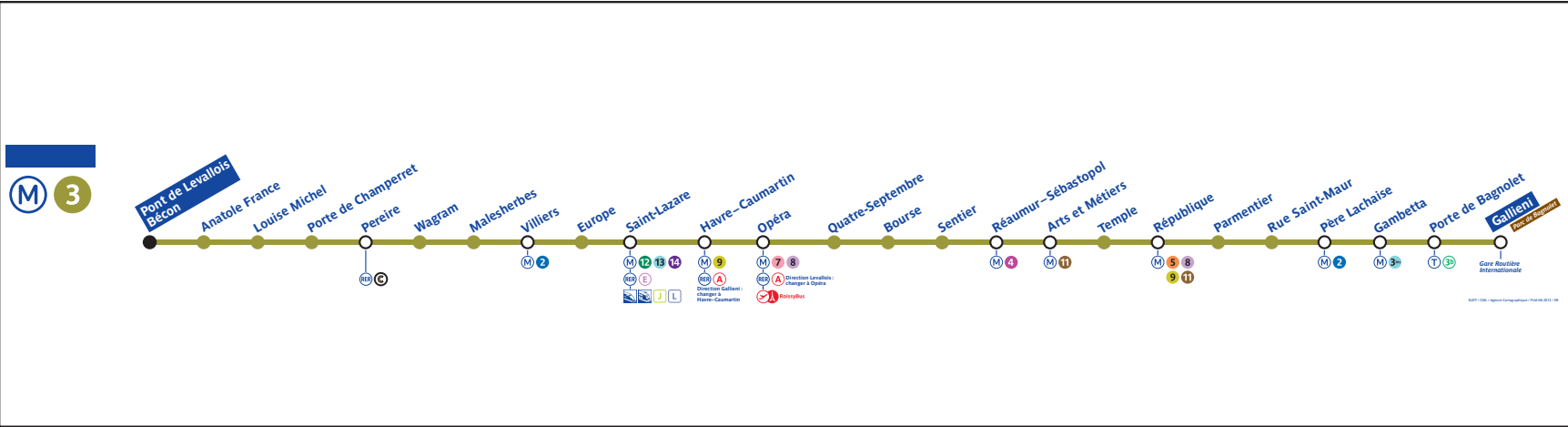


M 1

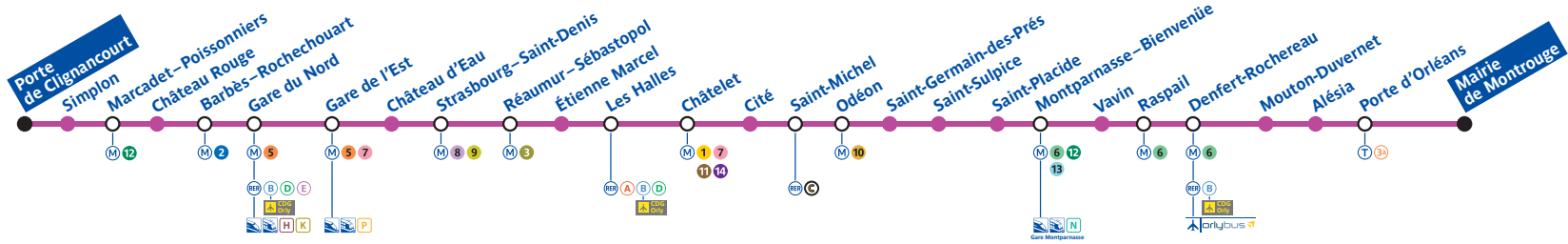


M 2



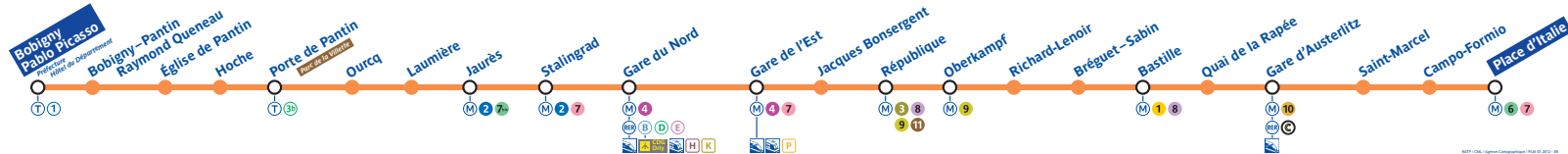


**M 4**



Métro - Carte d'Agence Cartographie - Mars 2012 - 40

**M 5**



Métro - Carte d'Agence Cartographie - Mars 2012 - 40

